# STPC
# Graphics Engine Driver Writer's Guide

Issue 1.1

September 1999



**STMicroelectronics**

Technoparc du Pays de Gex - B.P. 112
165, rue Edouard Branly
01630 Saint Genis Pouilly (France)

# 1. INTRODUCTION

## 1.1. 2D ACCELERATION

The STPC Graphics Engine provides enhanced 2D accelerations inside the Frame Buffer area of STPC Unified Memory Architecture. Frame Buffer area is dedicated to graphic and video data shared by CRTC (screen display), Video Pipeline (video overlay) and Video Input Port. Any of this data can be processed by the Graphics Engine as well as data stored in Frame Buffer by the CPU or an external hardware through the PCI interface.

The Graphics Engine processes high speed transfer of 2D rectangular areas with the possibility of additionnal processing of the data. By extension, the Graphics Engine can be used to process horizontal and vertical line drawing (one pixel width rectangle) or any geometric shape which can be split into such types of line: filled circles or polygons for example.

The source data processed by the Graphics Engine can be taken from the Frame Buffer, be it constant (fill a rectangle area) or come from a FIFO called the Data Port. This FIFO is filled by the CPU and used when the source data are not directly accessible (outside the Frame Buffer) or when the operation to be done on each pixels is not directly supported by Graphics Engine. In this second case, the CPU reads data, processes it and sends it to the Data Port to be written to the Frame Buffer taking advantage of the Graphics Engine faster write access. The Graphics Engine destination pixels are always located inside the Frame Buffer.

In place of a full image, a colored 8x8 pixel pattern can be used to fill the destination area.

In addition to data transfer, the Graphics Engine can process different types of operations :

- color key based transparency on source and destination data
- 8x8 pixels pattern transparency mask
- logical operations between source, destination and an 8x8 pixel patterns
- pixel conversion from monochrome 1 bit per pixel to colored format (usually for text display)

These additionnal operations are supported when the destination data follows one of these formats : 8, 16, 24 or 32 bits sized format and can be mixed to build more complex operations.

## 1.2. HARDWARE CURSOR

The STPC Graphics Engine provides a black and white hardware cursor up to 64x64 pixels. This cursor is managed as an independent graphics plan and doesn't interact with Frame Buffer's stored data. This allows the software to modify Frame Buffer data without having to take care of redrawing the cursor and display the cursor without having to save the background data.

# 2. GRAPHICS ENGINE 2D ACCELERATIONS

A graphic acceleration is setup in the following steps:

- Setup the data source type
- Setup the 8x8 pattern (optional)
- Setup the raster operation
- Setup transparency options (optional)
- Setup source and destination locations
- Setup transfer direction
- Start the acceleration

The type of graphic acceleration is determined by the data sources. Destinations are always a 8, 16, 24 or 32 bit pixel format rectangle located inside the Frame Buffer.

## 2.1. SETUP THE DATA SOURCE

Graphic acceleration can use and mix three data sources. Pixels from the destination area, pixels from an 8x8 pattern and pixels from a source image.

This chapter explain how to set up the source image.

The source image is defined by these parameters :

- Origin of source data
- Pixel format of source data
- Way source data is stored in the memory

## 2.2. ORIGIN OF SOURCE IMAGE

Sources data can be separate into 3 categories :

- The source is a constant color : used to fill a rectangle
- The source is a Frame Buffer rectangle : screen to screen Bitblit
- The source is from the Data Port FIFO : host to screen Bitblit

Origin of data is selected by bits 30 and 31 of ROP Register :

- 00 : Constant color
- 01 : Frame Buffer
- 10 : Data Port FIFO

When the source is a constant color, content of the Background Register is used and replicated as constant source data. In 8, 16 or 24 bit pixel format, only the 8, 16 or 24 low bits of this register are used.

When the source is the Frame Buffer, the source location has to be set up as describe in Section 2.7.

When source is the Data Port FIFO, the CPU has to send the source image to the Data Port Register or any address inside the 4Mb Data Port memory space. Data has to be sent 32 bits at a time and fast CPU string operation (movsd assembly langage function) can be used with the 4Mb memory space as destination. X values of Src_XY Register are set to the number of Bytes (bits when the source is 1 bit per pixel) to discard inside the first 32 bits of each line sent to the Data Port. This feature is used to align a CPU read access on 32 bits and accelerates the transfer.

## 2.3. SOURCE IMAGE FORMAT

Source pixels format can be :

- 1 bit per pixels
- Same as destination pixel format

The 1 bit per pixel source data is selected by setting the bitmap expansion bit of ROP register (bit 10) to 1. When selected, the Background Color register value is used to replace a 0 value and Foreground Color register to replace a 1 value. Every bit of source data is used to generate one output pixel (32 pixels stored in a 32 bit word). In 8, 16 or 24 bit destination pixel format, only 8, 16 or 24 low bits of this register are used.

In both bitmap expanded or normal mode, Pixel Depth register is used to select the number of Bytes per pixels of destination data. In not expanded mode, the source and destination pixel format must be the same.

The pixel format corresponds to the number of Bytes for a pixel : 1 Byte for 256 colors surfaces, 2 Bytes for 32K colors or 64K colors surfaces and 3 or 4 Bytes for 16M colors surfaces. 32K colors pixel format correspond to a 2 Byte per pixel format with one bit unused and is available only for compatibility with VESA standard. A 4 Byte per pixel format is a 16M colors mode with one Byte unused. 16 color graphic mode is not supported by the graphic accelerator.

### 2.3.1 The way the source image is stored in memory

The source data can be stored in normal or packed mode.

In packed mode, the image data is stored in the Frame Buffer (or send to the Data Port FIFO in case of host to screen) with no loss of space between two lines. Data of a line is directly followed by data of next line on a Byte basis for a color source image and on a bits basis for a 1 bit per pixel source image.

In normal mode, a pitch value is used to determine the position of the next line. This point is describe in detail in Section 2.7. In case of host to screen, unused data of the last 32 bits of each line are discarded on contrary to packed mode where this data is used as first data of the next line.

To enable packed mode in a normal Bitblit operation, the source pitch value should be set to 0. This value is none 0 when not packed mode is required.

To enable packed mode when source is 1 bit per pixel (bit expansion enabled) the packed mode bit of ROP register should be set to 1. The last bit of a line is directly followed by the first bit of the next line.

Packed or not packed mode have, of course, no effect when source image is a constant color.

## 2.4. 8X8 PATTERN

A Second useable source is an 8x8 pixel pattern. This pattern is always stored inside the Frame Buffer and follows these rules :

- pattern data is 256 Bytes aligned inside the Frame Buffer
- pattern data uses the same pixel color depth as the destination area. This pixel depth is determined by Pixel Depth register (8, 16, 24 or 32 bits per pixels).
- the number of Bytes between two lines of the pattern is fixed to 32 regardless of the pixel color depth.

To enable the use of a pattern, bit 28 of ROP register must be set to 1 and the location of the pattern in frame before must be set in pattern register. In the specific case of 24 bits per pixel, additionnal information has to be set in bits 3 and 4 of Pattern register : (DstX / 8) % 3 as describe in the data book (DstX correspond to the X coordinate of destination rectangle).

This pattern can be used :

- to be mixed with source image and destination in a raster operation.
- to be used as a mask for transparency.

These two possibilities are described in the next chapters.

## 2.5. CONFIGURE RASTER OPERATION

For every type of graphics acceleration, the user has to set up the raster operation in lower 8 bits of ROP register. This parameter defines the type of logical operation to mix the source, pattern and destination pixels. This 8 bit value follows the Windows standard :

Sample of ROP value

0xCCh : destination = source

0xEEh : destination = source OR destination

0x88h : destination = source AND destination

0x66h : destination = source XOR destination

0x44h : destination = source AND (NOT destination)

0x33h : destination = (NOT source)

0x11h : destination = (NOT src) AND (NOT destination)

0xC0h : destination = (source AND pattern)

0xBBh : destination = (NOT source) OR destination

0xF0h : destination = pattern

0x5Ah : destination = pattern XOR destination

0x55h : destination = (NOT destination)

0x00h : destination = BLACK

0xFFh : destination = WHITE

If destination data is used in the raster operation, bit 28 of the ROP register has to be set to 1.

If a pattern is used in the raster operation, bit 29 of ROP register has to be set to 1.

## 2.6. TRANSPARENCY OPERATION

During any data transfer, the STPC Graphics Engine can process different types of transparencies. These features are compatible with most of graphic accelerations described previously by using additionnal registers. STPC Graphics Engine provide 4 types of transparencies : source color compare, destination color compare, monochrome transparency and pattern based transparency.

### 2.6.1 Source color compare transparency

When this feature is enabled, the Graphics Engine compares the source data color with the SRC transparency compare register and writes the data to destination surface only if the color matches (or don't match) the reference color. This mode is enabled by setting bit 13 of ROP register to 1. If bit 12 is set to 1, only pixels which match the reference color will be transfered, if bit 12 is set to 0, pixels which don't match the value will be transfered.

'The source does not match' transparency mode is commonly used to display a picture with a transparent background.

### 2.6.2 Destination color compare transparency

When this feature is enabled, the Graphics Engine compare destination data color with the DST Transparency Compare register and writes the sources data to destination surface only if color match (or don't match) the reference color. This mode is enabled by setting bit 15 of ROP register to 1. If bit 14 is set to 1, sources pixels are written only if the color in the destination surface matches the reference color, if bit 14 is set to 0, sources pixels are written only if the color in the destination surface doesn't match the reference color.

When this mode is used, bit 28 of ROP register has to be set to 1 to allow the Graphics Engine to read destination data for the comparison.

### 2.6.3 Monochrome transparency

When the source is 1 bit per pixel, monochrome transparency bit of ROP register can be set to 1 to transfer only foreground pixels (value 1) from the source. Background pixels (value 0) will be ignored.

### 2.6.4 Pattern transparency

In this mode, an 8x8 pattern is used as a transparency mask. Source pixels are written to the destination surface only if the pattern value is non-zero and not written if the pattern value is 0. To enable this mode, bit 9 and 29 of the ROP register have to be set to 1 and the pattern register configured with the offset of the pattern in the Frame Buffer as describe in Section 2.4.

## 2.7. SOURCE AND DESTINATION LOCATION

To facilitate setup of the source and destination coordinates in the Frame Buffer, this operation is separated into two step :

- setup the Frame Buffer working area
- setup coordinates inside the working area

This way concerns only data located in the Frame Buffer. This means the destination rectangle and the source rectangle for a screen to screen Bitblit and only the destination rectangle for a fill operation or a host-to-screen biblit.

### 2.7.1 Setup working area

Working area is setup by two parameters :

- a base address
- a pitch

Base address is a 32 Bytes aligned offset in the Frame Buffer which correspond to the offset of the first pixel of the working area.

The pitch is the number of Bytes between the first pixel of two consecutives lines during the biblit operation. STPC Graphics Engine supports limited value for pitch defined by the sum of 4 values of $2^n$ values as describe in the databook.

Before the operation starts, the Destination Base and Pitch registers and eventually Source Base and Pitch registers have to be configured. When source data is in packed mode, the pitch is set to 0 as describe in Section 2.3.1

### 2.7.2 Source and destination coordinate

This value corresponds to the X and Y coordinates inside the working area. The first pixel processed by the Graphics Engine follow this formula :

First pixel Frame Buffer address = base + Y * pitch + X

The first pixel of the next line is defined by adding or substracting the pitch values, depending on operation direction as describe in Section 2.8.

The X coordinate corresponds to a value in number of Bytes and not in number of pixels. This means the X value in pixels has to be multiplied by the number of Bytes per pixels (value set in the Pixel Depth register).

### 2.7.3 Width and height

The Width and Height register have to be set to the size of the rectangle to be processed. When width or height values are fixed like for a line drawing operation (fixed to one pixel), a width-specified or height-specified Bitblit can be used in place of write width or height in the register. This feature is described in Section 2.9.2 & Section 2.9.3.

The width value correspond to a value in number of Bytes. This means the width value in pixels have to be multiply by the number of Byte per pixels (value set in pixel depth register).

Warning : Width and Height registers contain the real value minus 1. For width, the formula is :

(X * number of Byte per pixels) - 1.

## 2.8. TRANSFER DIRECTION

3 bits of pixel depth register control the Biblit operation direction. A normal operation is setup to do the copy from left to right and top to bottom and bit 5, 6 and 7 of Pixel Depth register have to be set to 0.

The transfer direction needs to be modified in two case :

- to provide a vertical flip during a screen to screen operation
- to avoid problems of overlap during a screen to screen operation

### 2.8.1 Screen to screen with vertical flip

This feature can be enabled by setting a different direction for the source and destination Y in Pixel Depth register (bits 6 and 7 = 0 and 1 or 1 and 0).

Warning : when inverting source (or destination) direction, the Y coordinate of source (or destination) needs to be set to the bottom of the processed rectangle in place of the top.

### 2.8.2 Avoid overlaping during screen to screen Bitblit

When destination rectangle overlaps the source rectangle in a screen to screen Bitblit operation, the copy direction needs to be setup carefully to be sure the Graphics Engine will not take data previously written inside Frame Buffer as source for the rest of the operation.

This means :

Horizontaly : if the Y source is equal to the Y destination the problem appears when the X destination is inside the source rectangle (X destination between X source and X source + width). In this case, a copy needs to be done from right to left in place of left to right : set bit 5 of the Pixel Depth register to 1 and set source and destination X coordinate to the right of the rectangle.

Verticaly : the problem appears when the Y destination is inside the source rectangle (Y destination between Y source and Y source + height). In this case, a copy needs to be done from bottom to top in place of top to bottom : set bit 6 and 7 of the Pixel Depth register to 1 and set source and destination Y coordinate to bottom of the rectangle.

Warning : when inverting the source (or destination) Y direction, the Y coordinate of source (or destination) needs to be set to the bottom of the rectangle to transfer in place of the top. When inverting the X direction, the X coordinate of source and destination needs to be set to the right of the rectangle to transfer in place of the left.

## 2.9. START THE GRAPHIC ACCELERATION

### 2.9.1 Start a standard operation

The graphic acceleration automaticly starts after setting up the Destination X and Y register so this register has to be the last one to be written. In case of host to screen Bitblit, the user has to send the source data to the Data Port FIFO after setting the Destination X, Y register.

The Destination X, Y register is visible in multiple location in the address space. For a standard Biblit operation, any 32 bits of the simple BitBlt area can be used as Destination XY register (offset 0x10000h to 0x13FFFh).

The Source and Destination Coordinate, the Width and the Height registers are double buffered and a second graphics operation can be set up before the end of the last one. If the user tries to intialize a third operation by accessing one of these registers, the CPU will be halted waiting for the end of the first operation. The availability of double buffered registers can be checked by reading bit 30 of Status register (pending busy bit).

This is not the case of the other Graphics Engine registers and it is more efficient to configure a type of acceleration once and start a series of operation just modifying the coordinate and size registers.

Warning : If one of the not double-buffered register is accessed during a Graphics Engine operation, the current operation will be corrupt. This means before modifying a none double buffered register, the user needs to check if Graphics Engine is not busy. This could be done by reading bit 31 of Status register.

### 2.9.2 Width specified Bitblit

In place of the Setup Width register, a Bitblit operation can be started using a Width Specified Destination XY register. This operation will save time when the widths are fixed like for vertical line (width = number of Bytes per pixel).

The Destination X, Y register is visible in multiple location in the address space. To start a width specified Bitblit, the Destination XY register to be used is one of the registers of the width specified area (0x14000h to 0x17FFFh).

The width used depends on the addressed used (bits 2 to 13 of the address = width minus 1) :

offset 0x14000h for a 1 Byte width line

offset 0x14004h for a 2 Bytes width line (1 pixel width in 2 Bytes per pixel mode)

offset 0x14008h for a 3 Bytes width line (1 pixel width in 3 Bytes per pixel mode)

offset 0x1400Ch for a 4 Bytes width line (1 pixel width in 4 Bytes per pixel mode)

## 2.9.3 Height specified Bitblit

In place of Setup Height register, a Bitblit operation can be started using a Height Specified Destination XY register. This operation will save time when the height is fixed like for an horizontal line (height = 1).

The Destination X, Y register is visible in multiple location in the address space. To start height specified Bitblit, Destination XY register to be used is one of the registers of the height specified area (0x18000h to 0x1BFFFh).

The height used depends on the address used (bits 2 to 13 of the address = height minus 1) :

offset 0x18000h for a 1 pixel height line

offset 0x18004h for a 2 pixels height line

offset 0x18008h for a 3 pixels height line

offset 0x1800Ch for a 4 pixels height line

# 3. HOW TO USE GRAPHICS ENGINE TO...

## 3.1. DISPLAY TEXT

Text can be displayed using a Bitblit with bitmap expansion. The Graphics Engine will automatically convert 1 bit per pixel format to the screen pixel format. The bitmap transparency option can be used to display text without erasing the background.

## 3.2. VERTICAL FLIP IMAGE DURING A BITBLIT

This can be done only for a screen to screen Bitblit by modifying the direction bit of Pixel Depth register. This method is explain in Section 2.8.

## 3.3. FILL A RECTANGLE WITH A PATTERN

This can be done using an 8x8 pattern and a 0xF0h ROP value. Source X and Y values will be used to determine the starting point inside the pattern.

## 3.4. DRAW HORIZONTAL AND VERTICAL LINES

Horizontal and vertical lines can be considered as width specified or height specified rectangular fills. Pattern transparency features can be used to display different line styles (dot, dash-dot...) as long as the repeated pattern is 8 pixels long.

## 3.5. FILL POLYGONS, CIRCLE OR A RANDOM GEOMETRIC SHAPE

Geometric shapes can be split into a set of horizontal lines and any type of acceleration can be used to fill the geometric shape (pattern, transparecy...). For each new line, only destination coordinates and widths have to be set up and the geometric shape drawing can take advantage of the double-buffering of these registers

## 3.6. MODIFICATIONS ON PIXELS DURING A BITBLIT

If a pixel modification is not supported by the Graphics Engine, it has to be provided during the Bitblit like a pixel format conversion (8, 16 or 24 Bytes per pixels to 8, 16 or 24 Bytes per pixels), the Graphics Engine can be used to accelerate the writing operation using an host to screen type of acceleration.

The CPU initializes a host to screen Bitblit, read source pixels from memory, modify them and send the modified pixels to the Data Port FIFO.

# 4. HARDWARE MOUSE CURSOR

## 4.1. SELECT THE MOUSE CURSOR COLORS

The two hardware cursor colors can be setup via CRTC registers 0x2Ah to 0x2Fh using 24 bit RGB values. These colors are always 24 bit RGB values, regardless of the graphics mode used.

## 4.2. SET THE CURSOR SHAPE

The cursor shape is described by atwo bit mask of 512 Bytes. The first mask has to be 1024 Bytes aligned in the Frame Buffer and the second mask just follows the first one in memory. These masks are composed of 64 lines of 8 Bytes (64 bits) to represente a 64x64 cursor.

The first mask determines whether a pixel is transparent (1) or not transparent (0) and the second mask if the pixel uses the first color (0 - usually black) or the second color (1 - usually white). The second mask value has to be put to 0 for transparency mode (when first mask bit is set to 1).

The width of the cursor is automatically 64 pixels wide and right colum has to be put to transparent if the cursor width is inferior. The cursor height is set up by register 0x29h of the CRTC and the offset of the two cursor masks by register 0x30h, 0x31h and 0x32h of the CRTC.

## 4.3. CHANGE THE CURSOR LOCATION ON SCREEN

The location of the upper left corner of the cursor shape is set up in the Cursor XY Graphics Engine register. The x and y values are only positive values and the Graphics Engine doesn't manage clipping of the cursor in the top or the left of the screen when the hot spot of the cursor is not in the upper left corner. In this case, the software builds a new clipped cursor and temporarily swaps the cursor shape with the clipped one.

# 5. GLOSSARY

Frame Buffer :

The Frame Buffer is an area of memory reserved for graphics and video. This area can be set between 128Kb to 4Mb of memory and the Graphics Engine can work only in this area. Frame Buffer memory is equivalent to a video memory in a none UMA architecture.

Host memory :

Host memory correspond to the system memory, visible by the operationg system.

Data Port :

The Data Port FIFO is used to send data to the Graphics Engine when this data is not directly accessible. The Data Port FIFO is mapped in memory and is accessible using a physical memory address in the Graphics Engine area.

Pixel depth :

The pixel depth is the number of Bytes for one pixel of a graphics area. The pixel depth defines the number of colors managed in this area.

Unified Memory architecture :

Unified Memory Architecture (UMA) is a memory architecture where system memory, graphics memory and video memory is physically grouped inside same memory chip, using the same memory bus.

CRTC :

The Cathodic Ray Tube Controller get data from the Frame Buffer to display them on a monitor or a TV.

Video Pipeline :

Video overlay used to display video on screen.

Video Input Port :

Video input used to capture a video stream inside the Frame Buffer.

Bitblit :

Operation of image (rectangle area) copied inside memory.

# 6. UPDATE HISTORY FOR GRAPHICS ENGINE WRIGHTER'S GUIDE

The following changes have been made to the Graphics Engine Wrighter's Guide on 04/11/99.

| Section | Change | Text |
|---|---|---|
| 2.4. | Removed | "~~Coordinate of the first used pixel in the pattern is set in Src_XY when this register is not used to describe the source rectangle (when source data is constant). This option can be used to center the pattern correctly in the destination area to fill.~~" |
| 3.3. | Removed | "~~Source X and Y values will be used to determine the starting point inside the pattern.~~" |